Deep Learning avec TensorFlow

MISE EN ŒUVRE ET CAS CONCRETS

Aurélien Géron

Traduit de l'anglais par Hervé Soulard

DUNOD

Authorized French translation of material from the English edition of
Hands-On Machine Learning with Scikit-Learn and TensorFlow,
ISBN 9781491962299
© 2017 Aurélien Géron.
This translation is published and sold by permission of O'Reilly Media, Inc.,
which owns or controls all rights to publish and sell the same.

Conception de la couverture : Randy Comer
Illustratrice : Rebecca Demarest

© Dunod, 2017
11 rue Paul Bert, 92240 Malakoff
www.dunod.com
ISBN 978-2-10-076747-2

Le Code de la propriété intellectuelle n'autorisant, aux termes de l'article L. 122-5, 2° et 3° a), d'une part, que les « copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective » et, d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration, « toute représentation ou reproduction intégrale ou partielle faite sans le consentement de l'auteur ou de ses ayants droit ou ayants cause est illicite » (art. L. 122-4).
Cette représentation ou reproduction, par quelque procédé que ce soit, constituerait donc une contrefaçon sanctionnée par les articles L. 335-2 et suivants du Code de la propriété intellectuelle.

Table des matières

Avant-propos VII

Chapitre 1. – Les fondamentaux du Machine Learning 1

1.1 Installer TensorFlow. 2

1.2 Installer le projet Handson-ML 3

1.3 Qu'est-ce que le Machine Learning ?. 5

1.4 Comment le système apprend-il ? 7

1.5 Régression linéaire 8

1.6 Descente de gradient 13

1.7 Régression polynomiale 24

1.8 Courbes d'apprentissage. 26

1.9 Modèles linéaires régularisés 30

1.10 Régression logistique 38

1.11 Exercices 46

Chapitre 2. – Introduction à TensorFlow 47

2.1 Créer un premier graphe et l'exécuter dans une session 50

2.2 Gérer des graphes. 52

2.3 Cycle de vie de la valeur d'un nœud 52

2.4 Manipuler des matrices 53

2.5 Régression linéaire avec TensorFlow. 54

2.6 Descente de gradient avec TensorFlow 56

2.7 Fournir des données à l'algorithme d'entraînement 58

2.8 Enregistrer et restaurer des modèles 60
2.9 Visualiser le graphe et les courbes d'apprentissage avec TensorBoard 61
2.10 Portées de noms 64
2.11 Modularité 65
2.12 Partager des variables 67
2.13 Exercices 70

Chapitre 3. – Introduction aux réseaux de neurones artificiels 73
3.1 Du biologique à l'artificiel 74
3.2 Entraîner un PMC avec une API TensorFlow de haut niveau 83
3.3 Entraîner un PMC avec TensorFlow de base 85
3.4 Régler précisément les hyperparamètres d'un réseau de neurones 91
3.5 Exercices 94

Chapitre 4. – Entraînement de réseaux de neurones profonds 97
4.1 Problèmes de disparition et d'explosion des gradients 98
4.2 Réutiliser des couches préentraînées 109
4.3 Optimiseurs plus rapides 118
4.4 Éviter le surajustement grâce à la régularisation 127
4.5 Recommandations pratiques 135
4.6 Exercices 136

Chapitre 5. – Distribution de TensorFlow sur des processeurs ou des serveurs 139
5.1 Plusieurs processeurs sur une seule machine 140
5.2 Plusieurs processeurs sur plusieurs serveurs 150
5.3 Paralléliser des réseaux de neurones dans une partition TensorFlow 167
5.4 Exercices 177

Chapitre 6. – Réseaux de neurones convolutifs 179
6.1 L'architecture du cortex visuel 180
6.2 Couche de convolution 181
6.3 Couche de *pooling* 189
6.4 Architectures de CNN 191
6.5 Exercices 203

Chapitre 7. – Réseaux de neurones récurrents 205
7.1 Neurones récurrents 206
7.2 RNR de base avec TensorFlow 210
7.3 Entraîner des RNR 215
7.4 RNR profonds 223
7.5 Cellule LSTM 227
7.6 Cellule GRU 230
7.7 Traitement automatique du langage naturel 231
7.8 Exercices 236

Chapitre 8. – Autoencodeurs 239
8.1 Représentations efficaces des données 240
8.2 ACP avec un autoencodeur linéaire sous-complet 241
8.3 Autoencodeurs empilés 243
8.4 Préentraînement non supervisé avec des autoencodeurs empilés 251
8.5 Autoencodeurs débruiteurs 252
8.6 Autoencodeurs épars 254
8.7 Autoencodeurs variationnels 256
8.8 Autres autoencodeurs 260
8.9 Exercices 261

Chapitre 9. – Apprentissage par renforcement 265
9.1 Apprendre à optimiser les récompenses 266
9.2 Recherche de politique 268
9.3 Introduction à OpenAI Gym 269
9.4 Politiques par réseau de neurones 272
9.5 Évaluer des actions : le problème d'affectation de crédit 274
9.6 Gradients de politique 276
9.7 Processus de décision markoviens 281
9.8 Apprentissage par différence temporelle et apprentissage Q 284
9.9 Apprendre à jouer à Ms. Pac-Man avec l'algorithme DQN de DeepMind 288
9.10 Exercices 296

© Dunod – Toute reproduction non autorisée est un délit.

Le mot de la fin 299

Annexe A. – Solutions des exercices 301

Annexe B. – Différentiation automatique 323

Annexe C. – Autres architectures de RNA répandues 331

Index 339

En prolongement de cet ouvrage, Aurélien Géron a mis en ligne une vidéo sur les réseaux de capsules, nouvelle architecture de réseaux neuronaux.

Cette vidéo est accessible à l'adresse suivante :
https://youtu.be/BJBxae3c1H4



Avant-propos

L'intelligence artificielle en pleine explosion

Auriez-vous cru, il y a seulement 10 ans, que vous pourriez aujourd'hui poser toutes sortes de questions à voix haute à votre téléphone, et qu'il réponde correctement ? Que des voitures autonomes sillonneraient déjà les rues (surtout américaines, pour l'instant) ? Qu'un logiciel, AlphaGo, parviendrait à vaincre Ke Jie, le champion du monde du jeu de go, alors que, jusqu'alors, aucune machine n'était jamais arrivée à la cheville d'un grand maître de ce jeu ? Que chaque jour vous utiliseriez des dizaines d'applications intelligentes, des outils de recherche à la traduction automatique en passant par les systèmes de recommandations ?

Au rythme où vont les choses, on peut se demander ce qui sera possible dans 10 ans ! Les docteurs feront-ils appel à des intelligences artificielles (IA) pour les assister dans leurs diagnostics ? Les jeunes écouteront-ils des tubes personnalisés, composés spécialement pour eux par des machines analysant leurs habitudes, leurs goûts et leurs réactions ? Des robots pleins d'empathie tiendront-ils compagnie aux personnes âgées ? Quels sont vos pronostics ? Notez-les bien et rendez-vous dans 10 ans ! Une chose est sûre : le monde ressemble de plus en plus à un roman de science-fiction.

L'apprentissage automatique se démocratise

Au cœur de ces avancées extraordinaires se trouve le Machine Learning (ML, ou *apprentissage automatique*) : des systèmes informatiques capables d'apprendre à partir d'exemples. Bien que le ML existe depuis plus de 50 ans, il n'a véritablement pris son envol que depuis une dizaine d'années, d'abord dans les laboratoires de recherche, puis très vite chez les géants du web, notamment les GAFA (Google, Apple, Facebook et Amazon).

À présent, le Machine Learning envahit les entreprises de toutes tailles. Il les aide à analyser des volumes importants de données et à en extraire les informations les plus utiles (*data mining*). Il peut aussi détecter automatiquement les anomalies de production, repérer les tentatives de fraude, segmenter une base de clients afin de mieux cibler les offres, prévoir les ventes (ou toute autre série temporelle), classer automatiquement les prospects à appeler en priorité, optimiser le nombre de conseillers de

clientèle en fonction de la date, de l'heure et de mille autres paramètres, etc. La liste d'applications s'agrandit de jour en jour.

Cette diffusion rapide du Machine Learning est rendue possible en particulier par trois facteurs :

- Les entreprises sont pour la plupart passées au numérique depuis longtemps : elles ont ainsi des masses de données facilement disponibles, à la fois en interne et *via* Internet.
- La puissance de calcul considérable nécessaire pour l'apprentissage automatique est désormais à la portée de tous les budgets, en partie grâce à la loi de Moore[1], et en partie grâce à l'industrie du jeu vidéo : en effet, grâce à la production de masse de cartes graphiques puissantes, on peut aujourd'hui acheter pour un prix d'environ 1 000 € une carte graphique équipée d'un processeur GPU capable de réaliser des milliers de milliards de calculs par seconde[2]. En l'an 2000, le superordinateur ASCI White d'IBM avait déjà une puissance comparable… mais il avait coûté 110 millions de dollars ! Et bien sûr, si vous ne souhaitez pas investir dans du matériel, vous pouvez facilement louer des machines virtuelles dans le cloud.
- Enfin, grâce à l'ouverture grandissante de la communauté scientifique, toutes les découvertes sont disponibles quasi instantanément pour le monde entier, notamment sur *https://arxiv.org*. Dans combien d'autres domaines peut-on voir une idée scientifique publiée puis utilisée massivement en entreprise la même année ? À cela s'ajoute une ouverture comparable chez les GAFA : chacun s'efforce de devancer l'autre en matière de publication de logiciels libres, en partie pour dorer son image de marque, en partie pour que ses outils dominent et que ses solutions de cloud soient ainsi préférées, et, qui sait, peut-être aussi par altruisme (il n'est pas interdit de rêver). Il y a donc pléthore de logiciels libres d'excellente qualité pour le Machine Learning.

Dans ce livre, nous utiliserons TensorFlow, développé par Google et rendu open source fin 2015. Il s'agit d'un outil capable d'exécuter toutes sortes de calculs de façon distribuée, et particulièrement optimisé pour entraîner et exécuter des réseaux de neurones artificiels.

L'avènement des réseaux de neurones

Le Machine Learning repose sur un grand nombre d'outils, provenant de plusieurs domaines de recherche : notamment la théorie de l'optimisation, les statistiques, l'algèbre linéaire, la robotique, la génétique et bien sûr les neurosciences. Ces dernières ont inspiré les réseaux de neurones artificiels (RNA), des modèles simplifiés des réseaux de neurones biologiques qui composent votre cortex cérébral : c'était en 1943, il y a plus de 70 ans ! Après quelques années de tâtonnements, les chercheurs sont parvenus à leur faire apprendre diverses tâches, notamment de classification ou

1. Une loi vérifiée empiriquement depuis 50 ans et qui affirme que la puissance de calcul des processeurs double environ tous les 18 mois.
2. Par exemple, 11,8 téraFLOPS pour la carte GeForce GTX 1080 Ti de NVidia. Un téraFLOPS égale mille milliards de FLOPS. Un FLOPS est une opération à virgule flottante par seconde.

de régression (c'est-à-dire prévoir une valeur en fonction de plusieurs paramètres). Malheureusement, lorsqu'ils n'étaient composés que de quelques couches successives de neurones, les RNA ne semblaient capables d'apprendre que des tâches rudimentaires. Et lorsque l'on tentait de rajouter davantage de couches de neurones, on se heurtait à des problèmes en apparence insurmontables : d'une part, ces réseaux de neurones « profonds » exigeaient une puissance de calcul rédhibitoire pour l'époque, des quantités faramineuses de données, et surtout, ils s'arrêtaient obstinément d'apprendre après seulement quelques heures d'entraînement, sans que l'on sache pourquoi. Dépités, la plupart des chercheurs ont abandonné le *connexionnisme*, c'est-à-dire l'étude des réseaux de neurones, et se sont tournés vers d'autres techniques d'apprentissage automatique qui semblaient plus prometteuses, telles que les arbres de décision ou les machines à vecteurs de support (SVM).

Seuls quelques chercheurs particulièrement déterminés ont poursuivi leurs recherches : à la fin des années 1990, l'équipe de Yann Le Cun est parvenue à créer un réseau de neurones à convolution (CNN, ou ConvNet) capable d'apprendre à classer très efficacement des images de caractères manuscrits. Mais chat échaudé craint l'eau froide : il en fallait davantage pour que les réseaux de neurones ne reviennent en odeur de sainteté.

Enfin, une véritable révolution eut lieu en 2006 : Geoffrey Hinton et son équipe mirent au point une technique capable d'entraîner des réseaux de neurones profonds, et ils montrèrent que ceux-ci pouvaient apprendre à réaliser toutes sortes de tâches, bien au-delà de la classification d'images. L'apprentissage profond, ou *Deep Learning*, était né. Suite à cela, les progrès sont allés très vite, et, comme vous le verrez, la plupart des articles de recherche cités dans ce livre datent d'après 2010.

Objectif et approche

Pourquoi ce livre ? Quand je me suis mis au Machine Learning, j'ai trouvé plusieurs livres excellents, de même que des cours en ligne, des vidéos, des blogs, et bien d'autres ressources de grande qualité, mais j'ai été un peu frustré par le fait que le contenu était d'une part complètement éparpillé, et d'autre part généralement très théorique, et il était souvent très difficile de passer de la théorie à la pratique.

J'ai donc décidé d'écrire le livre *Hands-On Machine Learning with Scikit-Learn and TensorFlow* (ou HOML), avec pour objectif de couvrir les principaux domaines du Machine Learning, des simples modèles linéaires aux SVM en passant par les arbres de décision et les forêts aléatoires, et bien sûr aussi le Deep Learning et même l'apprentissage par renforcement (Reinforcement Learning, ou RL). Je voulais que le livre soit utile à n'importe quelle personne ayant un minimum d'expérience de programmation (si possible en Python[3]), et axer l'apprentissage autour de la pratique, avec de nombreux exemples de code. Vous retrouverez ainsi tous les exemples de code de ce livre sur *https://github.com/ageron/handson-ml*, sous la forme de notebooks Jupyter.

3. J'ai choisi le langage Python d'une part parce que c'est mon langage de prédilection, mais aussi parce qu'il est simple et concis, ce qui permet de remplir le livre de nombreux exemples de code. En outre, il s'agit actuellement du langage le plus utilisé en Machine Learning (avec le langage R).

© Dunod – Toute reproduction non autorisée est un délit.

Notes sur l'édition française

La première moitié de HOML est une introduction au Machine Learning, reposant sur la librairie Scikit-Learn. La seconde moitié est une introduction au Deep Learning, reposant sur la librairie TensorFlow. Dans l'édition française, ce livre a été scindé en deux :

- la première partie (chapitres 1 à 8) a été traduite dans le livre *Machine Learning avec Scikit-Learn*, aux éditions Dunod ;
- la seconde partie (chapitres 9 à 16) a été traduite dans le livre que vous tenez entre les mains, *Deep Learning avec TensorFlow*. Les chapitres ont été renumérotés de 2 à 9, et un nouveau chapitre 1 a été ajouté, reprenant les points essentiels de la première partie.

Prérequis

Bien que ce livre ait été écrit plus particulièrement pour les ingénieurs en informatique, il peut aussi intéresser toute personne sachant programmer et ayant quelques bases mathématiques. Il ne requiert aucune connaissance préalable sur le Machine Learning mais il suppose les prérequis suivants :

- vous devez avoir un minimum d'expérience de programmation ;
- sans forcément être un expert, vous devez connaître le langage Python, et si possible également ses librairies scientifiques, en particulier NumPy, Pandas et Matplotlib ;
- enfin, si vous voulez comprendre comment les algorithmes fonctionnent (ce qui n'est pas forcément indispensable, mais est tout de même très recommandé), vous devez avoir certaines bases en mathématiques dans les domaines suivants :
  - l'algèbre linéaire, notamment comprendre les vecteurs et les matrices (par exemple comment multiplier deux matrices, transposer ou inverser une matrice),
  - le calcul différentiel, notamment comprendre la notion de dérivée, de dérivée partielle, et savoir comment calculer la dérivée d'une fonction.

Si vous ne connaissez pas encore Python, il existe de nombreux tutoriels sur Internet, que je vous encourage à suivre : ce langage est très simple et s'apprend vite. En ce qui concerne les librairies scientifiques de Python et les bases mathématiques requises, le site *github.com/ageron/handson-ml* propose quelques tutoriels (en anglais) sous la forme de notebooks Jupyter. De nombreux tutoriels en français sont disponibles sur Internet. Le site *fr.khanacademy.org* est particulièrement recommandé pour les mathématiques.

Plan du livre

- Le chapitre 1 reprend les éléments du livre *Machine Learning avec Scikit-Learn* qui sont indispensables pour comprendre le Deep Learning. Il montre d'abord comment installer TensorFlow et le projet contenant les exemples de code du livre (ainsi que les librairies dont il dépend), puis il présente les bases du Machine Learning, comment entraîner divers modèles linéaires à l'aide de la descente de gradient, pour des tâches de régression et de classification, et il présente quelques techniques de régularisation.
- Le chapitre 2 introduit la librairie TensorFlow.

- Le chapitre 3 présente les réseaux de neurones et comment les mettre en œuvre avec TensorFlow.
- Le chapitre 4 montre comment résoudre les difficultés particulières que l'on rencontre avec les réseaux de neurones profonds.
- Le chapitre 5 explique comment distribuer des réseaux de neurones sur plusieurs processeurs, éventuellement sur plusieurs serveurs.
- Le chapitre 6 présente les réseaux de neurones à convolution, excellents pour analyser des images.
- Le chapitre 7 présente les réseaux de neurones récurrents, capables de traiter des séquences, telles que des séries temporelles ou les mots d'une phrase.
- Le chapitre 8 traite des autoencodeurs : ces réseaux de neurones sont capables d'apprendre, sans supervision, à détecter des motifs dans les données. Ils sont également très utiles pour générer de nouvelles données semblables à celles reçues en exemple (par exemple pour générer des images de visages).
- Le chapitre 9 aborde l'apprentissage par renforcement, dans lequel un agent apprend par tâtonnements au sein d'un environnement dans lequel il peut recevoir des récompenses ou des punitions. Nous construirons en particulier un agent capable d'apprendre à jouer tout seul à Ms. Pac-Man.

Conventions

Les conventions typographiques suivantes sont utilisées dans ce livre :

Italique

Indique un nouveau terme, une URL, une adresse email ou un nom de fichier.

`Largeur fixe`

Utilisé pour les exemples de code, ainsi qu'au sein du texte pour faire référence aux éléments d'un programme, tels que des instructions, des mots clés, des noms de variables, de fonctions, de base de données, de types de données ou encore de variables d'environnement.



Ce symbole indique une astuce ou une suggestion.



Ce symbole indique une précision ou une remarque générale.



Ce symbole indique une difficulté particulière ou un piège à éviter.

© Dunod – Toute reproduction non autorisée est un délit.

Remerciements

Je tiens tout d'abord à remercier mes collègues de Google, en particulier l'équipe de classification de vidéos YouTube, pour m'avoir tant appris sur l'apprentissage automatique. Je n'aurais jamais pu lancer ce projet sans eux. Un merci tout particulier à mes gourous ML personnels : Clément Courbet, Julien Dubois, Mathias Kende, Daniel Kitachewsky, James Pack, Alexander Pak, Anosh Raj, Vitor Sessak, Wiktor Tomczak, Ingrid von Glehn, Rich Washington et tout le monde à YouTube Paris.

Je suis extrêmement reconnaissant à toutes les personnes qui ont pris le temps d'examiner mon livre avec une grande attention. Merci à Pete Warden, membre de l'équipe TensorFlow, qui a répondu à toutes mes questions au sujet de TensorFlow, a relu les chapitres sur le Deep Learning et m'a apporté de nombreux éclairages. N'hésitez pas à aller voir son blog (*petewarden.com*), il en vaut la peine. Un grand merci aussi à Lukas Biewald, qui a relu les chapitres sur le Deep Learning en détail : aucun paragraphe n'a échappé à son regard rigoureux, il a testé tout le code (et permis de résoudre quelques erreurs), il a fait de nombreuses excellentes suggestions, et son enthousiasme a été particulièrement contagieux. Vous devriez également consulter son blog (*lukasbiewald.com*) et ses adorables petits robots (*goo.gl/Eu5u28*). Merci également à Justin Francis, qui a relu les chapitres sur le Deep Learning avec grande attention, découvrant quelques erreurs ici et là et apportant de nombreuses idées, en particulier dans le chapitre sur l'apprentissage par renforcement. N'hésitez pas à découvrir ses articles sur TensorFlow (*goo.gl/28ve8z*) ! Merci à Grégoire Mesnil, qui a relu les chapitres sur le Deep Learning et apporté de nombreuses suggestions pratiques sur la meilleure façon d'entraîner un réseau de neurones.

Concernant la première partie du livre original, un immense merci à David Andrzejewski, qui a relu et a fait des retours très utiles, en identifiant des sections à clarifier et en suggérant comment les améliorer. N'hésitez pas à visiter son site web (*david-andrzejewski.com*). Merci aussi à Eddy Hung, Salim Sémaoune, Karim Matrah, Ingrid von Glehn, Iain Smears et Vincent Guilbeau pour leur lecture attentive et leurs nombreuses suggestions.

Et je souhaite également remercier mon beau-père, Michel Tessier, ancien professeur de mathématiques et maintenant grand traducteur d'Anton Tchekhov, pour m'avoir aidé à éliminer certaines erreurs mathématiques et à clarifier les notations dans ce livre et dans les notebooks Jupyter.

Et bien sûr un immense merci à mon cher frère Sylvain, qui a passé en revue chaque chapitre, a testé toutes les lignes de code, a fourni des commentaires sur pratiquement toutes les sections et m'a encouragé de la première ligne à la dernière. *Love you, bro!*

Un grand merci au personnel fantastique d'O'Reilly, en particulier à Nicole Tache, qui m'a donné des commentaires perspicaces, toujours encourageants et utiles. Merci aussi à Marie Beaugureau, Ben Lorica, Mike Loukides et Laurel Ruma d'avoir cru en ce projet et de m'avoir aidé à le définir. Merci à Matt Hacker et à toute l'équipe d'Atlas pour avoir répondu à toutes mes questions techniques concernant le formatage asciidoc et LaTeX. Merci également à Rachel Monaghan, Nick Adams et à toute l'équipe de production pour leur relecture finale et leurs centaines de corrections.

Je souhaite également remercier Jean-Luc Blanc, des éditions Dunod, pour avoir soutenu et géré ce projet, et pour ses relectures attentives. Je tiens aussi à remercier vivement Hervé Soulard pour sa traduction. Enfin, je remercie chaleureusement Brice Martin, des éditions Dunod, pour sa relecture extrêmement rigoureuse, ses excellentes suggestions et ses très nombreuses corrections. Je lui dois même un double remerciement car il a également relu et corrigé avec le même souci du détail le livre *Machine Learning avec Scikit-Learn*, dans lequel j'ai malheureusement oublié de le remercier.

Pour finir, je suis infiniment reconnaissant à ma merveilleuse épouse, Emmanuelle, et à nos trois enfants, Alexandre, Rémi et Gabrielle, pour leur patience et leur soutien sans faille pendant l'écriture de ce livre. J'ai même eu droit à des biscuits et du café, comment rêver mieux ?

© Dunod – Toute reproduction non autorisée est un délit.

1

Les fondamentaux du Machine Learning

Avant de partir à l'assaut du mont Blanc, il faut être entraîné et bien équipé. De même, avant d'attaquer le Deep Learning avec TensorFlow, il est indispensable de maîtriser les bases du Machine Learning. Si vous avez lu le livre *Machine Learning avec Scikit-Learn* (A. Géron, Dunod, 2017), vous êtes prêt(e) à passer directement au chapitre suivant. Dans le cas contraire, ce chapitre vous donnera les bases indispensables pour la suite[4].

Nous commencerons par installer TensorFlow et les autres librairies Python dont vous aurez besoin pour exécuter les nombreux exemples de code. Dans ce premier chapitre nous utiliserons uniquement NumPy, Matplotlib et Scikit-Learn : nous attaquerons TensorFlow à partir du prochain chapitre.

Ensuite, nous étudierons la régression linéaire, l'une des techniques d'apprentissage automatique les plus simples qui soient. Cela nous permettra au passage de rappeller ce qu'est le Machine Learning, ainsi que le vocabulaire et les notations que nous emploierons tout au long de ce livre. Nous verrons deux façons très différentes d'entraîner un modèle de régression linéaire : premièrement, une méthode analytique qui trouve directement le modèle optimal (c'est-à-dire celui qui s'ajuste au mieux au jeu de données d'entraînement) ; deuxièmement, une méthode d'optimisation itérative appelée *descente de gradient* (en anglais, *gradient descent* ou GD), qui consiste à modifier graduellement les paramètres du modèle de façon à l'ajuster petit à petit au jeu de données d'entraînement.

Nous examinerons plusieurs variantes de cette méthode de descente de gradient que nous utiliserons à maintes reprises lorsque nous étudierons les réseaux de

4. Ce premier chapitre reprend en grande partie le chapitre 4 du livre *Machine Learning avec Scikit-Learn*, ainsi que quelques éléments essentiels des chapitres 1 à 3.

neurones artificiels : descente de gradient groupée (ou batch), descente de gradient par mini-lots (ou mini-batch) et descente de gradient stochastique.

Nous examinerons ensuite la régression polynomiale, un modèle plus complexe pouvant s'ajuster à des jeux de données non linéaires. Ce modèle ayant davantage de paramètres que la régression linéaire, il est plus enclin à surajuster (*overfit*, en anglais) le jeu d'entraînement. C'est pourquoi nous verrons comment détecter si c'est ou non le cas à l'aide de courbes d'apprentissage, puis nous examinerons plusieurs techniques de régularisation qui permettent de réduire le risque de surajustement du jeu d'entraînement.

Enfin, nous étudierons deux autres modèles qui sont couramment utilisés pour les tâches de classification : la régression logistique et la régression softmax.

Ces notions prises individuellement ne sont pas très compliquées, mais il y en a beaucoup à apprendre dans ce chapitre, et elles sont toutes indispensables pour la suite, alors accrochez-vous bien, c'est parti !

1.1 INSTALLER TENSORFLOW

Commençons par vérifier que vous disposez bien de Python 3. Pour cela, ouvrez un terminal et tapez la commande suivante (ne saisissez que ce qui se trouve après le `$`) :

```
$ python3 -V
Python 3.5.3
```

La version de Python doit s'afficher (dans cet exemple, c'est la version 3.5.3). Une version supérieure ou égale à 3.5 est recommandée, mais pas obligatoire. Vous pouvez éventuellement utiliser plutôt Python 2.7 (dans ce cas, utilisez la commande `python` plutôt que `python3`). Toutefois, Python 2 arrive en fin de vie et il est recommandé de passer à Python 3 dès que possible. Si nécessaire, installez donc la version de Python que vous préférez. Par la suite, nous supposerons que vous utilisez Python 3 (note : si vous utilisez une distribution spéciale de Python telle qu'Anaconda, vous devrez adapter les instructions suivantes à votre environnement).

Pour installer TensorFlow, nous utiliserons ici le logiciel pip : il s'agit du gestionnaire de librairies inclus avec Python. Commencez par vous assurer que vous possédez une version récente de pip :

```
$ pip3 install --user --upgrade pip
```

Remplacez la commande `pip3` par `pip` si vous utilisez Python 2. L'option `--user` indique à pip d'installer la mise à jour uniquement pour l'utilisateur actuel, ce qui a l'avantage de ne requérir aucun droit particulier. Si vous préférez une installation système (donc accessible à l'ensemble des utilisateurs de votre machine), vous devez enlever cette option, mais cela exigera alors sans doute les droits administrateur : sous macOS ou Linux, il vous faudra remplacer `pip3` par `sudo pip3`, et sous Windows il vous faudra ouvrir le terminal en mode administrateur. Cette remarque s'applique aussi aux commandes ci-dessous où l'option `--user` apparaît.

Si vous souhaitez installer TensorFlow dans un environnement Python bien isolé du système et des autres projets (ce qui permet à chacun de vos projets de disposer de

sa propre liste de librairies, potentiellement avec des versions de librairies distinctes), il vous suffit d'installer la librairie `virtualenv` et de l'utiliser pour créer un nouvel environnement Python, reposant sur la version de Python de votre choix :

```
$ pip3 install --user --upgrade virtualenv
$ mkdir $HOME/ml
$ cd $HOME/ml
$ virtualenv -p `which python3` env
```

Nous utilisons dans ce livre le répertoire de travail `$HOME/ml` mais vous pouvez le remplacer par tout autre répertoire de votre choix.

Désormais, lorsque vous souhaiterez utiliser cet environnement, il vous suffira d'ouvrir un terminal et de taper les commandes suivantes :

```
$ cd $HOME/ml
$ source env/bin/activate
```

Voilà ! Que vous ayez choisi de créer un environnement Python isolé ou non, vous êtes maintenant prêt(e) à installer TensorFlow. Pour cela, il suffit de lancer la commande suivante (attention : l'option `--user` doit être omise si vous utilisez un environnement isolé, et aucun droit d'accès particulier ne sera alors nécessaire) :

```
$ pip3 install --user --upgrade tensorflow
```



Si vous disposez d'un processeur GPU compatible avec TensorFlow, vous pouvez remplacer `tensorflow` par `tensorflow-gpu` (voir le chapitre 5 pour plus de détails).

Maintenant, vérifions que TensorFlow est bien installé :

```
$ python3 -c 'import tensorflow as tf; print(tf.__version__)'
1.3.0
```

Si la version de TensorFlow s'affiche (par exemple 1.3.0), alors bravo ! TensorFlow est correctement installé.

1.2 INSTALLER LE PROJET HANDSON-ML

Tous les exemples de code de ce livre, ainsi que du livre *Machine Learning avec Scikit-Learn*, sont disponibles sur GitHub dans le projet open source suivant :
https://github.com/ageron/handson-ml

Ce projet contient un notebook (bloc-note) Jupyter pour chaque chapitre. Jupyter est une interface web permettant d'exécuter du code Python (ou autres langages) de façon graphique et interactive. Les notebooks 1 à 8 correspondent aux chapitres 1 à 8 du livre *Machine Learning avec Scikit-Learn*, tandis que les notebooks 9 à 16 correspondent aux chapitres 2 à 9 du livre que vous tenez entre les mains (en effet, ces deux livres n'en font qu'un dans la version originale). Bien que ce soit facultatif, vous êtes fortement encouragé(e) à installer ce projet, à jouer avec le code, à faire les exercises pratiques et à regarder de près les corrections qui figurent à la fin de

© Dunod – Toute reproduction non autorisée est un délit.

chaque notebook : le Machine Learning et le Deep Learning s'apprennent surtout par la pratique !

Pour installer ce projet, il vous faudra le logiciel git. S'il n'est pas déjà présent sur votre système, suivez les instructions d'installation disponibles sur *https://git-scm.com/*. Ensuite, clonez le projet sur votre machine, à l'aide des commandes suivantes :

```
$ cd $HOME/ml
$ git clone https://github.com/ageron/handson-ml.git
```

Si vous avez créé un environnement isolé, vous devez maintenant l'activer (*cf.* plus haut). Puis, installez les librairies requises par ce projet. Cela inclut bien sûr Jupyter, ainsi que les librairies scientifiques les plus populaires en Python[5] (NumPy, SciPy, Pandas, Matplotlib), Scikit-Learn (qui met en œuvre de nombreux algorithmes de Machine Learning), et quelques autres.

```
$ cd handson-ml
$ pip3 install --user --upgrade -r requirements.txt
```

Comme expliqué plus haut, vous devez omettre l'option `--user` si vous utilisez un environnement isolé ou si vous souhaitez installer les librairies pour l'ensemble du système (pas juste pour l'utilisateur actuel). Vous pouvez enfin démarrer Jupyter :

```
$ jupyter notebook
```

Cette commande démarre le serveur web de Jupyter, et ouvre votre navigateur Internet sur l'adresse *http://localhost:8888/tree*. Vous devez y voir le contenu du répertoire courant. Il ne vous reste plus qu'à cliquer sur le notebook `04_training_linear_models.ipynb` : il s'agit du notebook correspondant au chapitre actuel (pour les chapitres suivants, rajoutez 7 au numéro du chapitre).

Si vous n'avez jamais utilisé Jupyter, le principe est simple : chaque notebook est constitué d'une liste de cellules. Chacune peut contenir du texte formaté ou du code (Python, dans notre cas). Lorsqu'on exécute une cellule de code, le résultat s'affiche sous la cellule. Cliquez sur le menu Help > User Interface Tour pour un tour rapide de l'interface. Pour vous entraîner, insérez quelques cellules de code au début du notebook, et exécutez quelques commandes Python, telles que `print("Hello world!")` (voir la figure 1.1). Vous pouvez renommer le notebook en cliquant sur son nom (voir la flèche 1 sur la figure). Cliquez dans une cellule de code et saisissez le code à exécuter (flèche 2), puis exécutez le code de la cellule en tapant Shift-Entrée ou en cliquant sur le bouton d'exécution (flèche 3). Lorsque vous cliquez à l'intérieur d'une cellule, vous passez en mode édition (la cellule est alors encadrée en vert). Lorsque vous tapez la touche Echap (Esc) ou que vous cliquez juste à gauche de la cellule, vous passez en mode commande (la cellule est alors encadrée en bleu). Lorsque vous êtes en mode commande, tapez la touche H pour afficher les nombreux raccourcis clavier disponibles.

5. Pour des tutoriels sur NumPy, Pandas et Matplotlib, voir les notebooks correspondants dans le projet.